A P P E N D I X

Copyright 1996, Paul Nelson

[illegible]

Time-Dimensional Tables

One of the biggest problems facing database developers is the representation of historic and future-dated information in the tables of a relational database. Typically, a type of table referred to as a "versioned table", which has an effective date as the last column of its primary key, is created to capture date-sensitive information. Although relational database management systems are capable of capturing date-sensitive information in its most basic form through the use of versioned tables, they are unable to adequately process such information. Retrieval of rows from a versioned table is a complex matter usually involving the use of subquery logic. Inexperienced requesters are unable to create such statements and a significant amount of development time is required of those requesters who are experienced. In addition, referential integrity cannot be appropriately enforced for versioned tables. In order to validate referential constraints, the DBMS currently matches the values of dependent table columns with the value of the primary key of the parent table. The value of the effective date column in the primary key of a versioned table represents only the first in a range of dates for which the information is effective. This range of dates is referred to as an "effective window". Referential constraints can be enforced for the value of the effective date, but not for the entire effective window. Because of these problems, database developers often avoid creating versioned tables to hold date-sensitive information. In many cases, business requirements which dictate the use of versioned tables are ignored. There have been many methods devised which avoid some of the problems inherent with versioned tables, but none of these methods fully address the problems. Some of the methods actually introduce additional problems. The DBMS could be modified to accommodate date-sensitive information in a way that would allow for simple retrieval against versioned tables and which would also enforce referential constraints on an effective window basis. This method termed "time-dimensional tables" is described in the remainder of this article.

Time-dimensional tables are defined to the database management system in much the same way that versioned tables are currently defined. The phrase "TABLE IS TIME-DIMENSIONAL" is included in the definition of a time-dimensional table. When the DBMS encounters this phrase, it marks the table as time-dimensional. The last column in the primary key is recognized by the DBMS as the effective start date column for the table. All other columns in the physical primary key form what will be referred to as the "logical primary key" of the table. The logical primary key should be recognized as the identifier for the information contained in a time-dimensional table. It is possible that several rows exist in a table with the same logical primary key. Each of these rows represent the same information at various points in time. The effective start date column allows these related rows to be differentiated. Thus, an additional dimension, the "time dimension" is recognized by the DBMS.

In many cases, a column is included in a versioned table to indicate the date on which a row is no longer effective. This would be specified during the definition of a time-dimensional table by including the phrase "EFFECTIVE END COLUMN IS column-name". The column named in the phrase cannot participate in the primary key of the table since it is not required to uniquely identify versions. The values of the effective start date column and the effective end date column are used by the DBMS to determine the effective window for each row of a time-dimensional table. The range of dates in the effective window of a row will be determined as all dates from (and including) the effective start date until (but not including) the effective end date. If a table has not been defined with an effective end date or a row contains a null entry in its effective end date column, the effective end date will be defaulted. First, the DBMS will attempt to retrieve next chronological row of the table with the same logical primary key. If a row is found, the effective start date of that row will be used as the default of the effective end date of the original row. By defaulting in this manner, gaps between the effective dates are prevented. If there are no rows with the same logical primary key and a later effective start date, the value of the effective end date is defaulted to the highest possible date value. A row is considered to be "active" on all dates that fall within its effective window. A row is considered to be "pending activation" on all dates prior to its effective window. A row is considered to be "inactive" on all dates following its effective window.

A predicate is added to the WHERE clause of the SQL to simplify requests for information from time-dimensional tables. The format of the new time-dimensional predicate is shown in table 1. The (table-name) clause is optional and identifies the time-dimensional table for which the predicate applies. If the (table-name) clause is omitted, the predicate will apply for all time dimensional tables referenced in a request. The (IS/WAS/WILL BE) clause is optional and is added merely for the sake of clarity. The ACTIVE/INACTIVE/PENDING clause is the only clause that is required and will be used by the DBMS to determine which rows of a time-dimensional table are to be included in the result set based upon their effective status. If ACTIVE is specified, all rows which represent current information on the target date will be included. If INACTIVE is specified, all rows which represent information that is no longer current on the target date will be included. If PENDING is specified, all rows which represent information that is not yet current on the target date will be included. The (NOT) clause is optional and indicates that the effective status specified in the ACTIVE/INACTIVE/PENDING clause is to be used to exclude rather than include rows. The (ON date) clause is optional and is used to specify a target date for time dimensional processing. The "(BETWEEN start-date AND end-date)" clause is also optional and is used to specify a range of target dates for time-dimensional processing. When this clause is included in the predicate, the range of target dates will be determined as all dates from (and including) the date specified in the start-date parameter until (and excluding) the date specified in the end-date parameter. The (ON date) clause and the (BETWEEN start-date AND end-date) clause may not be used in the same time-dimensional predicate. If neither target date specification clause is included, the current system date will be defaulted as the target date for the predicate.

TABLE 1
WHERE
(table-name)
(IS/WAS/WILL BE) (NOT) ACTIVE/INACTIVE/PENDING
(ON date) (BETWEEN start-date AND end-date)

Since the complex logic required to retrieve rows from time-dimensional tables is static and predictable, the function of creating this logic is internalized into the DBMS. By internalizing this logic, an opportunity for efficiency maximization is created in addition to relieving the requester of the complicated task. Only the most basic time-dimensional retrieval criteria must be included in a request. This basic information is passed to the DBMS from the requester through the use of the time-dimensional predicate. The DBMS expands the predicate using information defined for time-dimensional tables and completes the request.

TIME-DIMENSIONAL EXAMPLE 1 (retrieving current information)

Figure 1 shows two tables of an example database. Assume that the EMPL_PAY_RATE table has been defined as a time-dimensional table. The DBMS would recognize the PAY_EFF_DATE as the effective start date of the table since it is the last column in the physical primary key. The EMPL_ID column would be recognized as the table's logical primary key. The EMPL_PAY_RATE table can be used to maintain several different versions of an individual employee's pay rate. The different versions would be identified by the values of the PAY_EFF_DATE column. Historical, current and future pay rates may all be tracked in the EMPL_PAY_RATE table.

```
                              FIGURE 1
EMPLOYEE                       EMPL_PAY_RATE
    PK,EMPL_ID                     PK,EMPL_ID
    NAME                           PK,PAY_EFF_DATE
                                   PAY_RATE
```

Table 2 shows two requests which retrieve employee names and the current rate of pay for each employee from the example tables. The request titled "CURRENT METHOD" includes the complex subquery logic required to process the versioned tables. The request titled "TIME-DIMENSIONAL METHOD" uses the time-dimensional predicate to pass time-dimensional retrieval criteria to the DBMS. The predicate in this case requests that the DBMS only return rows which are currently active from the EMPL_PAY_RATE table. The DBMS will use the information previously defined for the time-dimensional table to fill in the missing pieces of the request. The result sets from both requests will be identical. The table name included in the time-dimensional predicate shown in table 2 is optional since the EMPL_PAY_RATE table is the only time-dimensional table referenced in the request.

TABLE 2

CURRENT METHOD:

```
SELECT NAME,
       PAY_RATE
FROM   EMPLOYEE       EMPL,
       EMPL_PAY_RATE PAY
WHERE  EMPL.EMPL_ID      = PAY.EMPL_ID
  AND  PAY.PAY_EFF_DATE =
      (SELECT MAX(PAY2.PAY_EFF_DATE)
       FROM   EMPL_PAY_RATE PAY2
       WHERE  PAY2.EMPL_ID       = EMPL.EMPL_ID
         AND  PAY2.PAY_EFF_DATE <= CURRENT DATE)
```

TIME-DIMENSIONAL METHOD:

```
SELECT NAME,
       PAY_RATE
FROM   EMPLOYEE       EMPL,
       EMPL_PAY_RATE PAY
WHERE  EMPL.EMPL_ID = PAY.EMPL_ID
  AND  EMPL_PAY_RATE IS ACTIVE
```

TIME-DIMENSIONAL EXAMPLE 2 (retrieving historic information using a date range)

Figure 2 shows two additional tables of the example database introduced in Figure 1. The EMPL_OCCUP table has been defined as a time-dimensional table with the OCCUP_EFF_DATE identified as its effective start date. The logical primary key of the EMPL_OCCUP table is the EMPL_ID column.

FIGURE 2

```
OCCUPATION                  EMPL_OCCUP
   PK,OCCUP_ID                 PK,EMPL_ID
   TITLE                       PK,OCCUP_EFF_DATE
                               OCCUP_ID
```

Table 3 shows two requests which retrieve employee names and identifies occupations worked by each employee anytime during the year 1994. The request titled "CURRENT METHOD" contains the complex subquery logic currently required to perform the task. The version titled "TIME-DIMENSIONAL METHOD" uses the time-dimensional predicate to pass time-dimensional criteria to the DBMS. The time-dimensional predicate shown specifies that the DBMS should only return rows from time-dimensional tables which were active between January 1st, 1994 and January 1st, 1995. The EMPL_OCCUP table is the only time-dimensional table referenced in the request, so the DBMS will assume that the predicate applies to that table only. The EMPL_OCCUP table definition will be accessed by the DBMS to determine other information required to complete the request.

TABLE 3

CURRENT METHOD:

```
SELECT NAME,
       OCCUP_ID
FROM   EMPLOYEE   EMPL,
       EMPL_OCCUP OCCUP
WHERE  EMPL.EMPL_ID             = OCCUP.EMPL_ID
  AND  OCCUP.OCCUP_EFF_DATE < '01/01/1995'
  AND (OCCUP.OCCUP_EFF_DATE >=
      (SELECT MAX(OCCUP2.OCCUP_EFF_DATE)
       FROM   EMPL_OCCUP OCCUP2
       WHERE  OCCUP2.EMPL_ID          = EMPL.EMPL_ID
         AND  OCCUP2.OCCUP_EFF_DATE <= '01/01/1994')
   OR  NOT EXISTS
      (SELECT MAX(OCCUP2.OCCUP_EFF_DATE)
       FROM   EMPL_OCCUP OCCUP2
       WHERE  OCCUP2.EMPL_ID          = EMPL.EMPL_ID
         AND  OCCUP2.OCCUP_EFF_DATE <= '01/01/1994'))
```

TIME-DIMENSIONAL METHOD:

```
SELECT NAME,
       OCCUP_ID
FROM   EMPLOYEE   EMPL,
       EMPL_OCCUP OCCUP
WHERE  EMPL.EMPL_ID = OCCUP.EMPL_ID
  AND  ACTIVE BETWEEN '01/01/1994' AND '01/01/1995'
```

TIME-DIMENSIONAL REFERENTIAL INTEGRITY

Exceptions must be made in order to enforce referential constraints for time-dimensional tables. In any case where the parent table in a referential constraint is time-dimensional, the columns defined in the constraint must correspond to the logical primary key of the parent table. In other words, the effective start date column of time-dimensional tables is not included in referential constraints. This represents a departure from the current method which enforces referential constraints based on the entire physical primary key of the parent table. When validating a time-dimensional referential constraint, any row in the parent table which has a logical primary key equal to the foreign key of the dependent row will satisfy the constraint. It is possible that several rows of the parent table would satisfy the constraint.

In many cases, it will be necessary to ensure that a time-dimensional parent row is active for a specific range of dates defined by columns of the dependent table. In this situation, the referential constraint definition would include the phrase "EFFECTIVE START DATE IS dependent-table-column". An effective end date may also be defined for a referential constraint by including the phrase "EFFECTIVE END DATE IS dependent-table-column" in its definition. An effective window for each dependent row is determined in the same manner that effective windows are defined for time-dimensional tables. In order for a referential constraint which includes effective dates to be satisfied, one or more parent table rows whose logical primary keys match the foreign key of the dependent table must be effective for the entire range of dates identified in the effective window.

CONCLUSION

The method of time-dimensional tables described in this article solves most of the problems that exist concerning the maintenance and retrieval of versioned information using relational and object-oriented database management systems. Individual tables would be capable of maintaining historical, active and pending versions of information. Referential constraints would be enforced in an appropriate manner for versioned information. The process of retrieving versioned information from a database would become relatively simple. Other features could also be included to enhance functionality. For instance, the DBMS could be modified to automatically prevent the effective windows of the rows of a versioned table from overlapping. The data types used for version control would not have to be limited to date data types. The effective start column and the effective end column of a versioned table could be of any data type which has an implied order. Data types which include a time, number data types, and some character data types could be used for version control.

If the method of time-dimensional tables described were to be implemented in a DBMS, the process of maintaining and retrieving versioned information might be simplified to the point that developers would no longer be afraid to include versioned information in their databases.